Basics Featuring Micro800 and Connected Components Workbench

Traffic Light Controller

For Classroom Use Only!

LISTEN. THINK. SOLVE.®

Allen-Bradley • Rockwell Software

Rockwell Automation

L4 — PLC Basics Featuring Micro800 and Connected Components Workbench

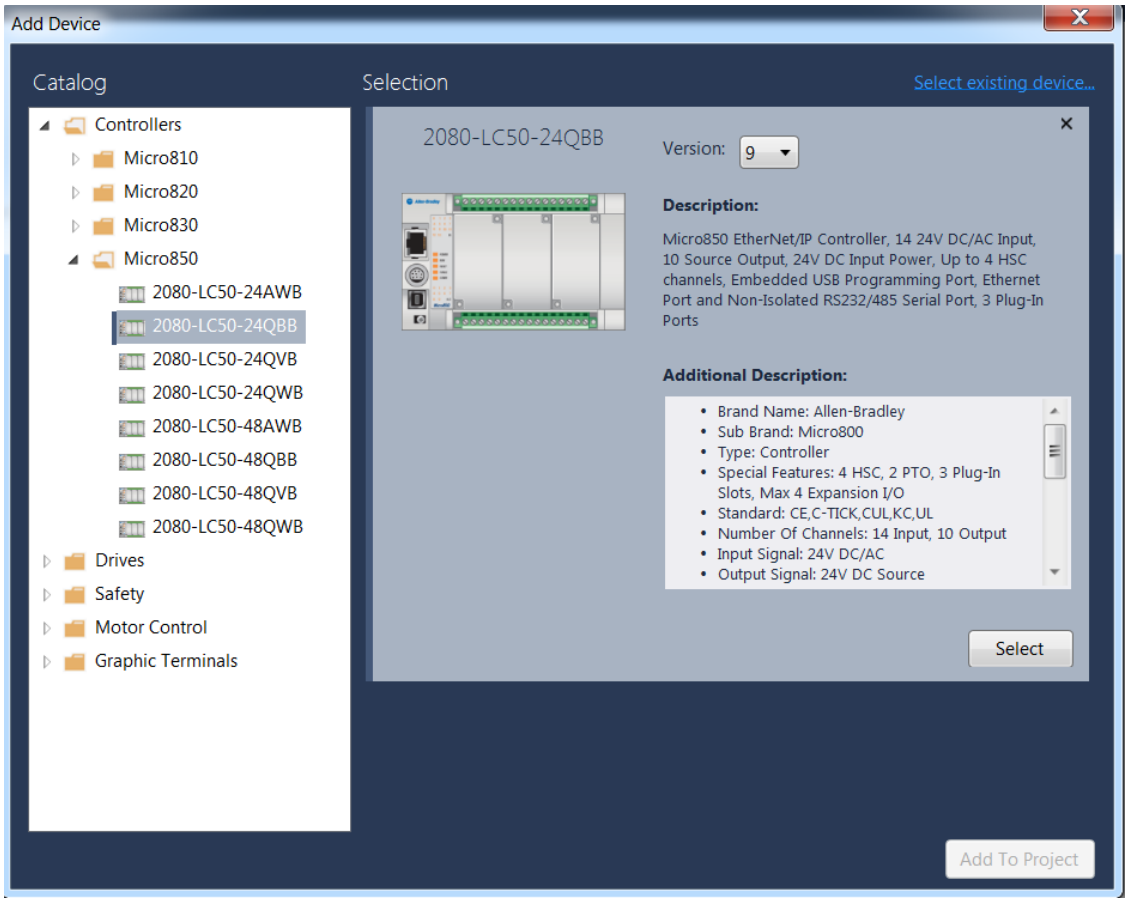
Programming a Micro800 using Connected Components Workbench

1. Connect your laptop to the M850 using the USB connection
2. Start the Connected Components Workbench software.

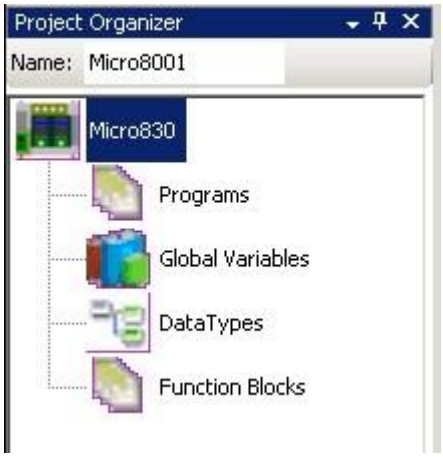
   Double click on the **CCW** icon 

   or from the **Start** menu, select **Programs > Rockwell Automation > CCW > CCW.Shell.exe**

3. Create a new project by clicking ‘File” – “New” or New Project if you’re on the splash screen.
4. Select the LC50-24QBB

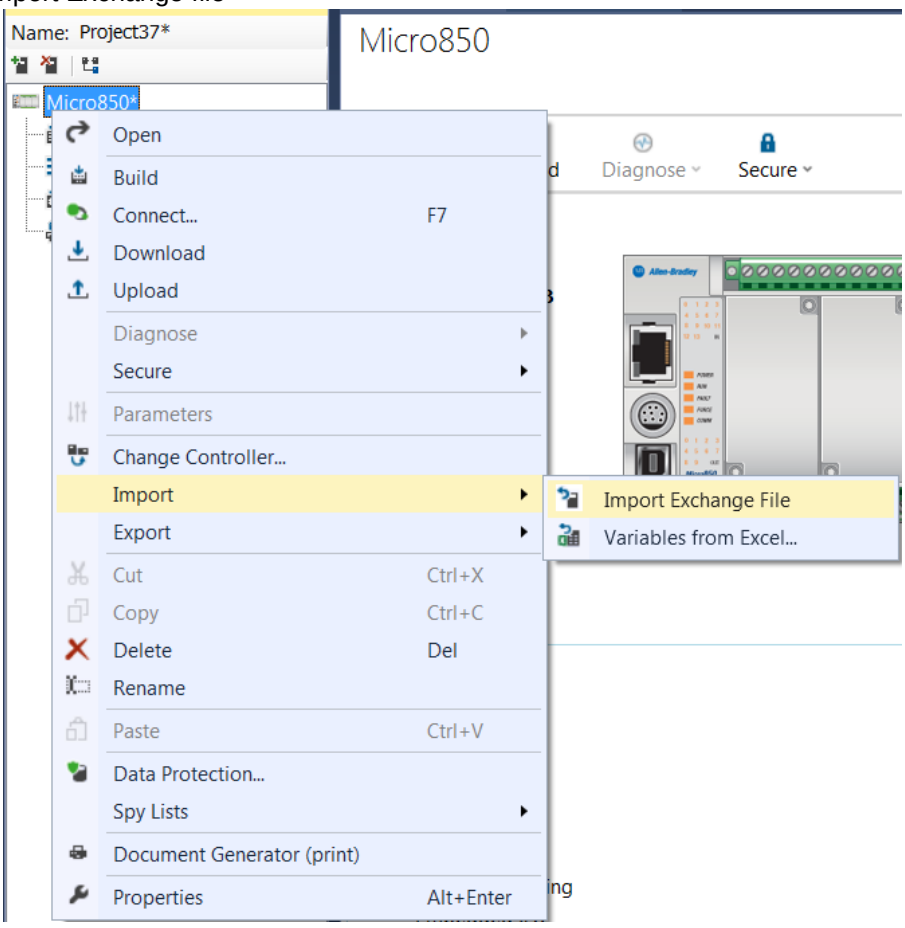


5. A new Micro850 project based on this controller has now been created. The Micro850 should show up in the **Project Organizer** on the left-hand side of the Workbench screen:

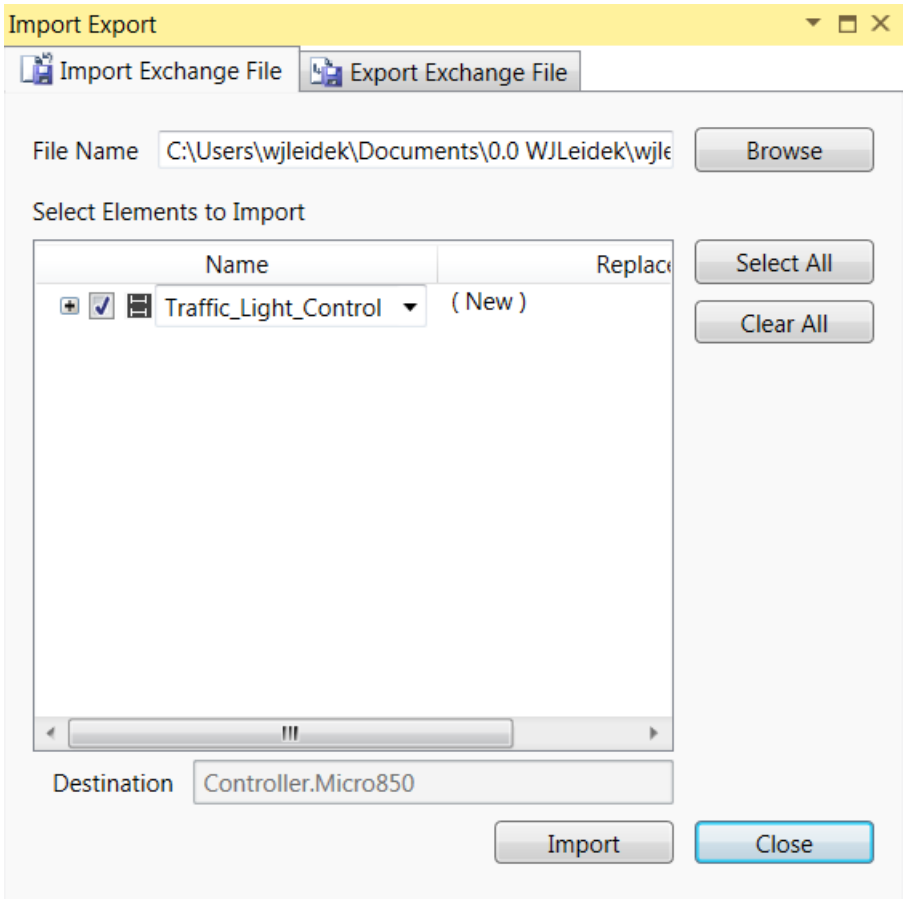


6. The goal for this lab is to program the Micro850 to control the traffic light at a typical 4-way intersection. The good news is that by taking advantage of a Traffic Controller User Defined Function Block (UDFB) that someone else has already created for us, it will be very straightforward to implement. The first thing we need to do is import this UDFB.

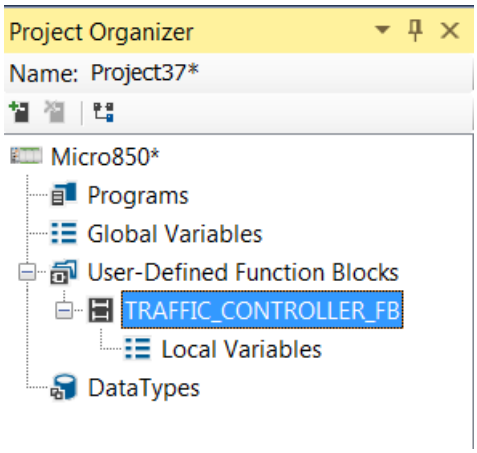
7. Under the Project Organizer, highlight the Controller, Right click and select “IMPORT’, then select Import Exchange file



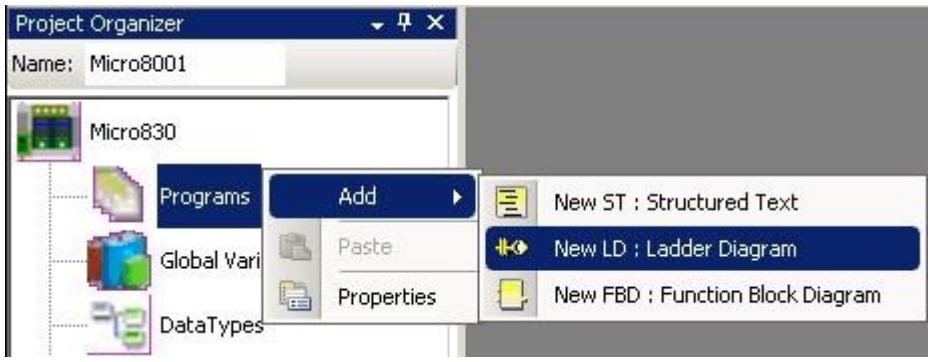
8. **Browse** to the location where the following file is stored controller.Micro830.Micro830.Traffic_Light_Control
Select it and click Import



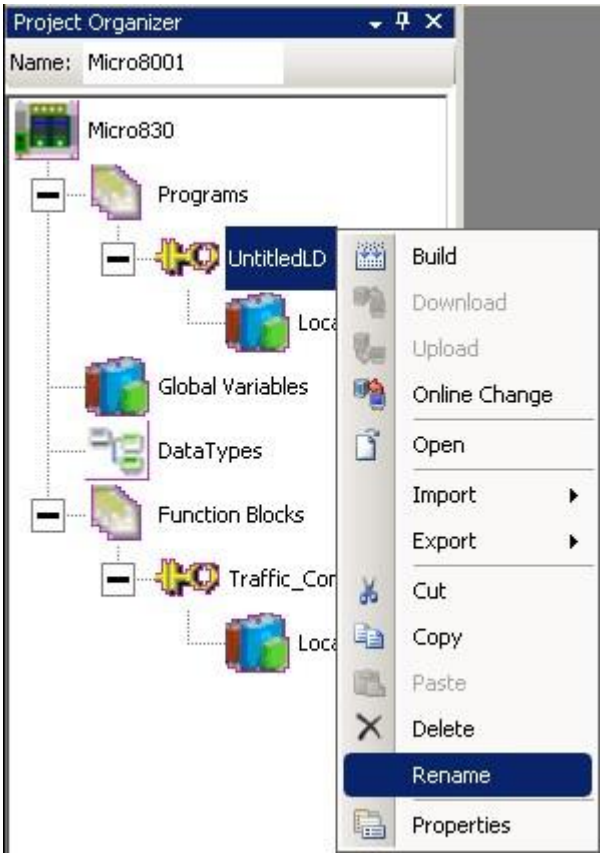
9. You should see an output message at the bottom indicating that the data was imported and you should see the **Traffic_Controller_FB** showing up under the Function Blocks within **Project Organizer**:



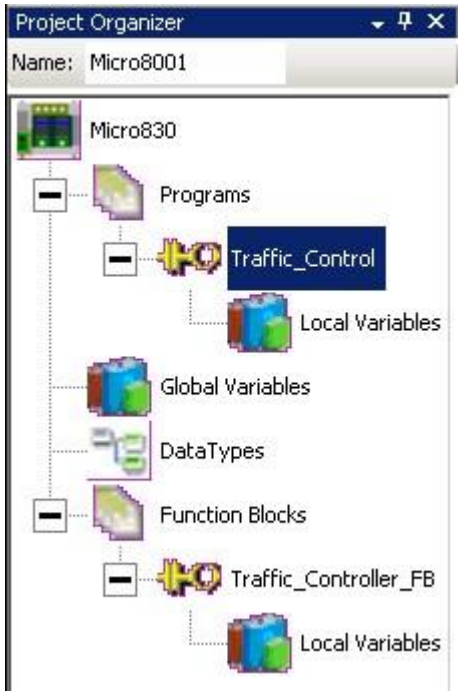
10. Under **Project Organizer**, right click on **Programs** select **Add** and select **New LD : Ladder Diagram**:



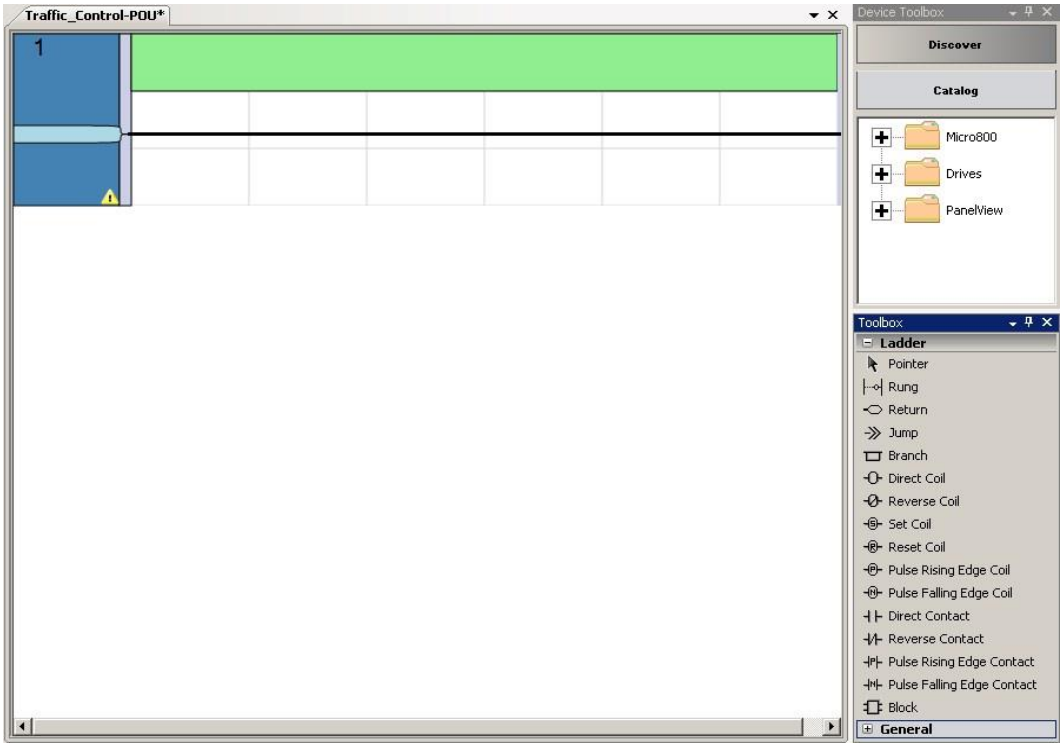
11. Right click on **UntitledLD** and select **Rename**:



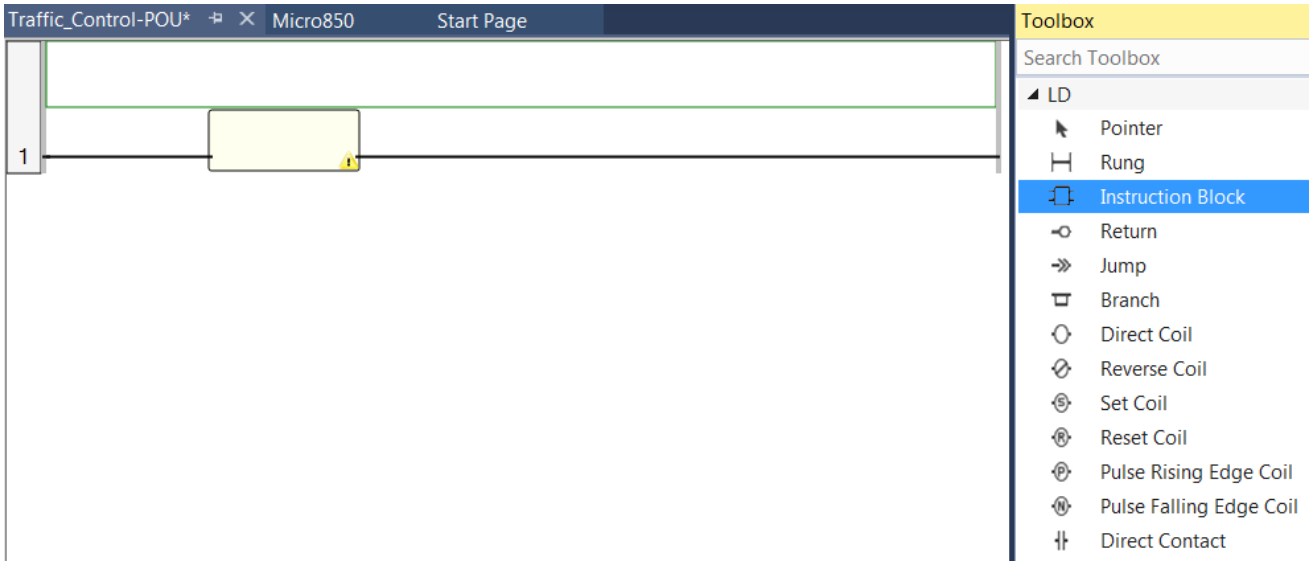
12. Type **Traffic Control** and Enter:



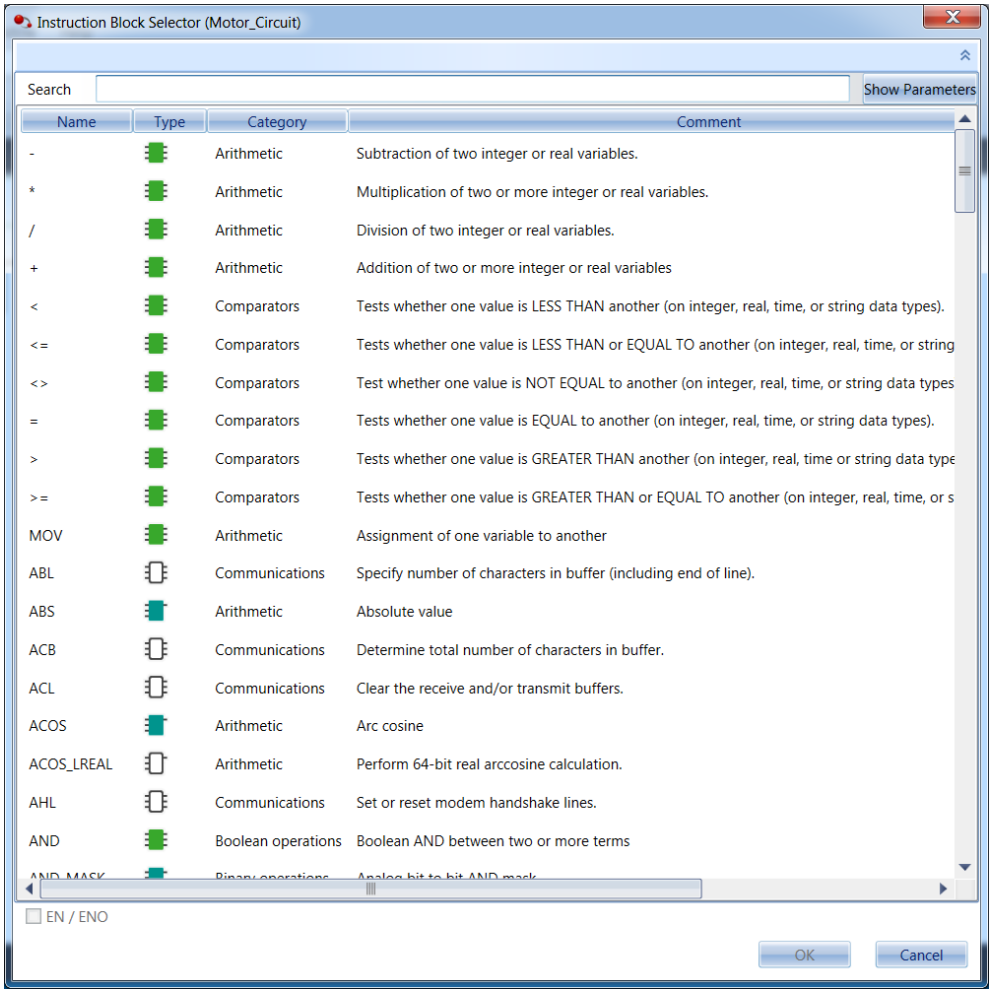
13. Double click on **Traffic Control** within **Project Organizer** to start editing the ladder logic program. Expand **Ladder** within the **Toolbox** in the lower left-hand screen to list the available instructions:



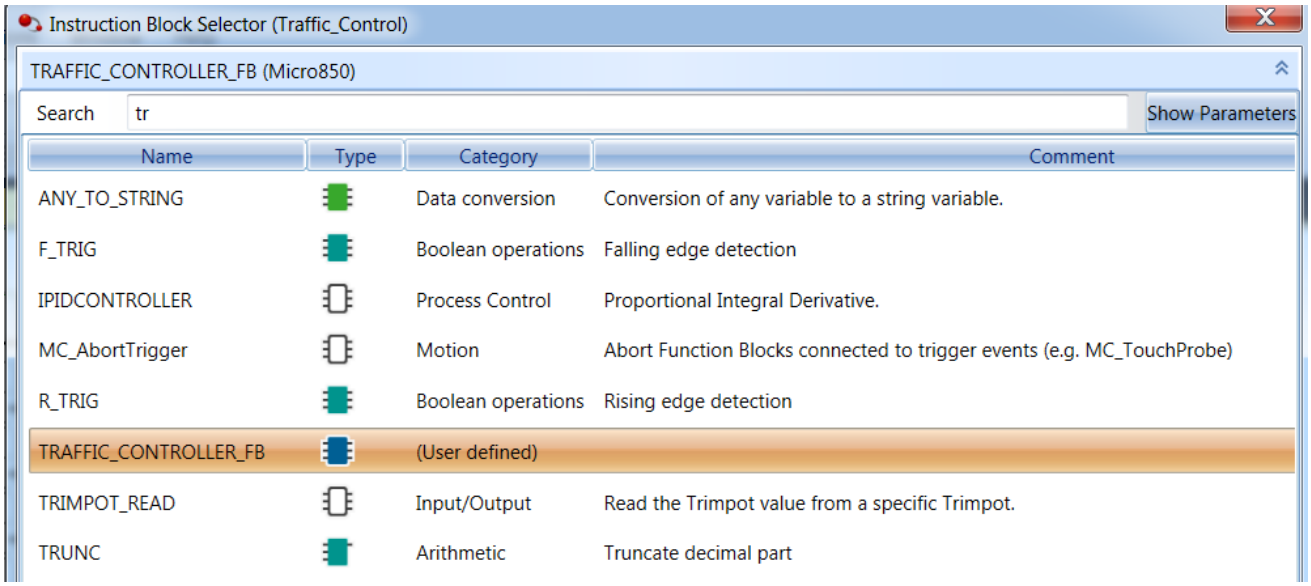
14. Click and hold on the **Block** instruction within the **Toolbox** and drag a Block onto the rung as shown:



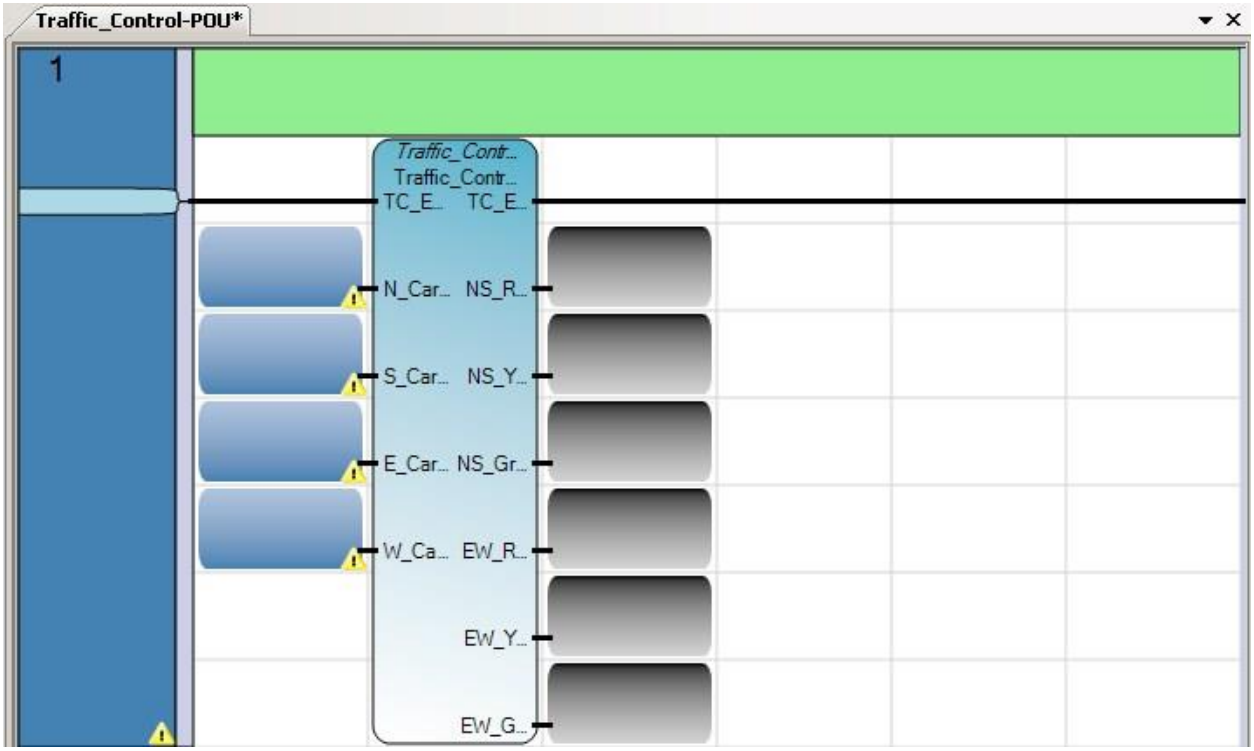
15. When you release the mouse button, the **Block Selector** screen appears:



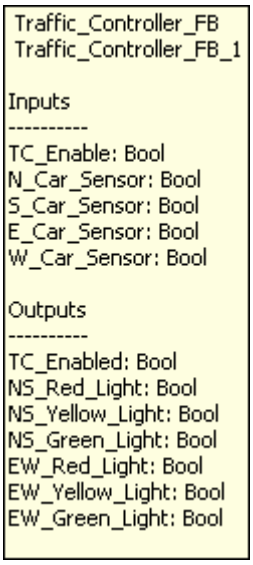
16. Under **Name**, type **tr** and note that only the instructions starting with tr are listed. Click on **Traffic_Controller_FB**



17. Select **Traffic_Controller_FB** in the list and click **OK**. The Traffic Controller function block should be displayed on the rung as shown below:

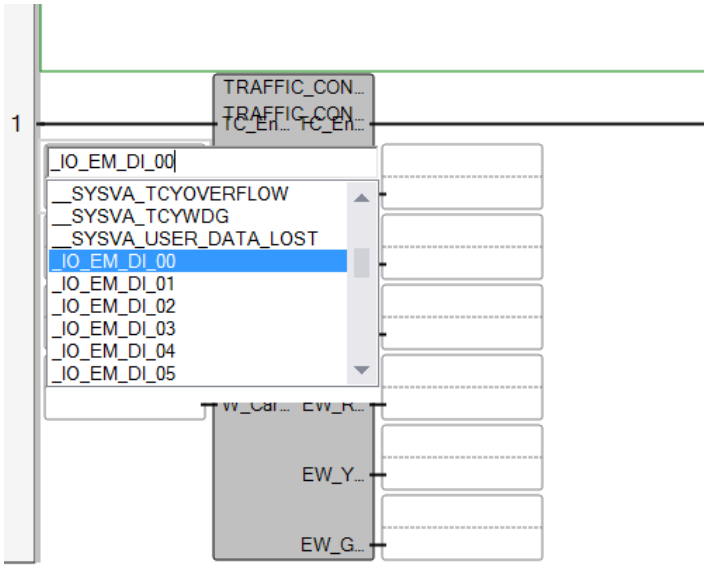


18. By convention, function blocks list inputs on the left-hand side of the block and outputs on the right-hand side of the block. In order to see the full names and data types of the variables that these inputs and outputs are associated with, move your cursor over the block - you should get the following listing:

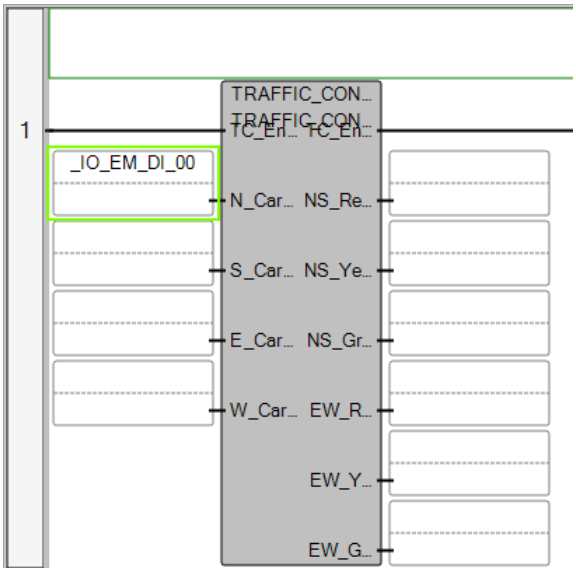


19. The first function block input that connects directly to the ladder rung is the function block enable bit. The remaining four Traffic Controller function block inputs are “real world” inputs that indicate whether a car is waiting at a red light in any of the four possible directions – North, South, East and West. These inputs get mapped to four Boolean input variables local to the function block: N_Car_Sensor, S_Car_Sensor, E_Car_Sensor and W_Car_Sensor. We are going to assign four Micro830 controller inputs to these function block inputs.

20. Click on the top of the input variable block that connects to **N_Car…** and you will get a dropdown menu of all the existing variable names that could be assigned to N_Car_Sensor. Scroll down and select **_IO_Embedded_Digital_Input_0** and enter:



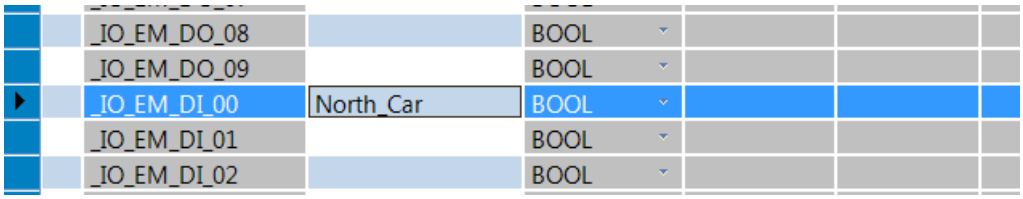
21. You should see something like this



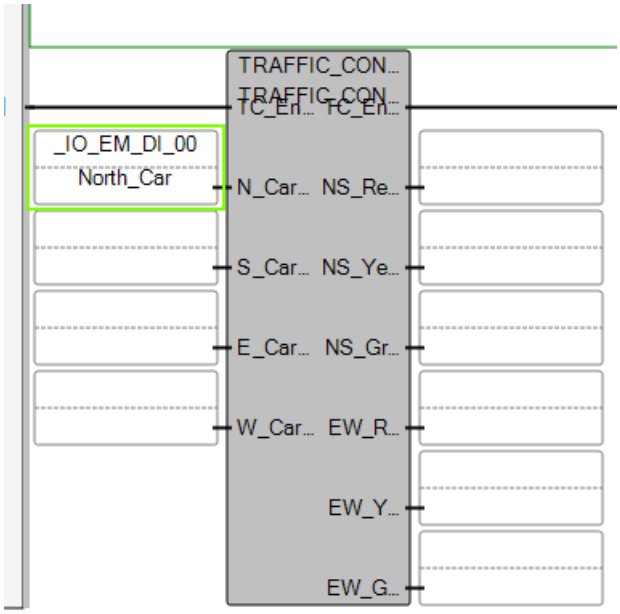
22. To simplify which part of the intersection this Input is- double click on the blank box below _IO_EM_DI_00
    This brings up the brings up the Global Tag Database

	Name	Alias	Data Type	Dimension	Project Val
	_IO_EM_DO_00		BOOL		
	_IO_EM_DO_01		BOOL		
	_IO_EM_DO_02		BOOL		
	_IO_EM_DO_03		BOOL		
	_IO_EM_DO_04		BOOL		
	_IO_EM_DO_05		BOOL		
	_IO_EM_DO_06		BOOL		
	_IO_EM_DO_07		BOOL		
	_IO_EM_DO_08		BOOL		
	_IO_EM_DO_09		BOOL		
▶	_IO_EM_DI_00		BOOL		
	_IO_EM_DI_01		BOOL		
	_IO_EM_DI_02		BOOL		

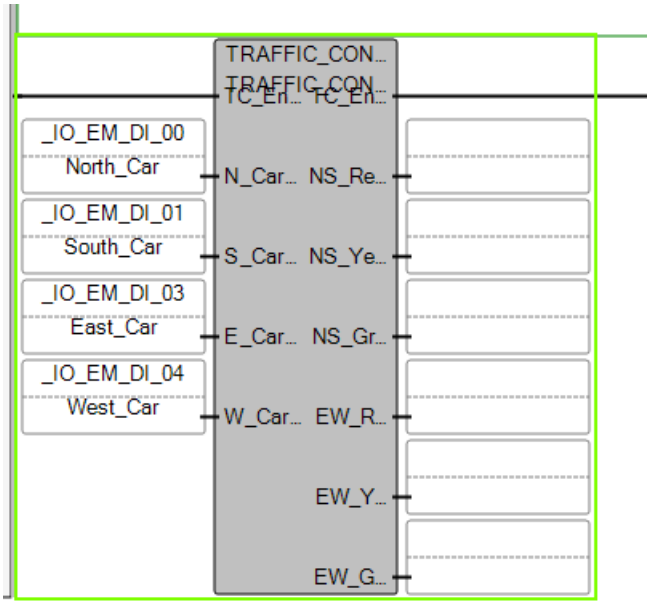
23. In the ‘Alias’ column, type in North_Car, and click OK.



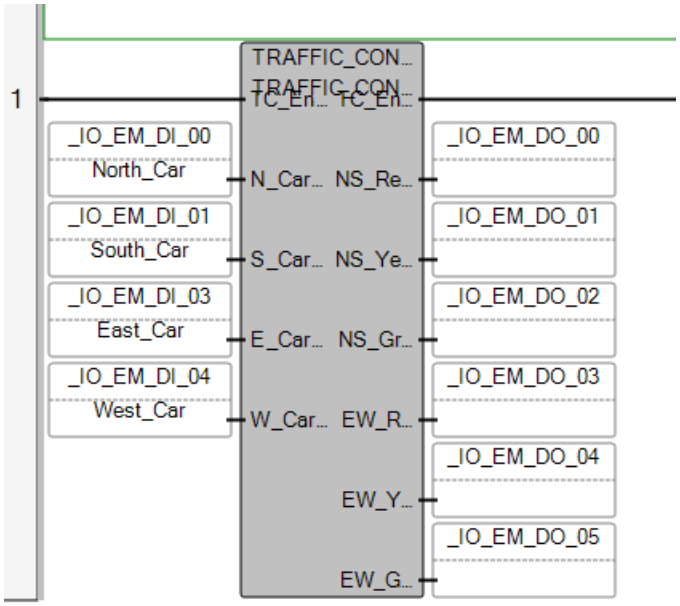
24. Your Function Block Input now contains easier information



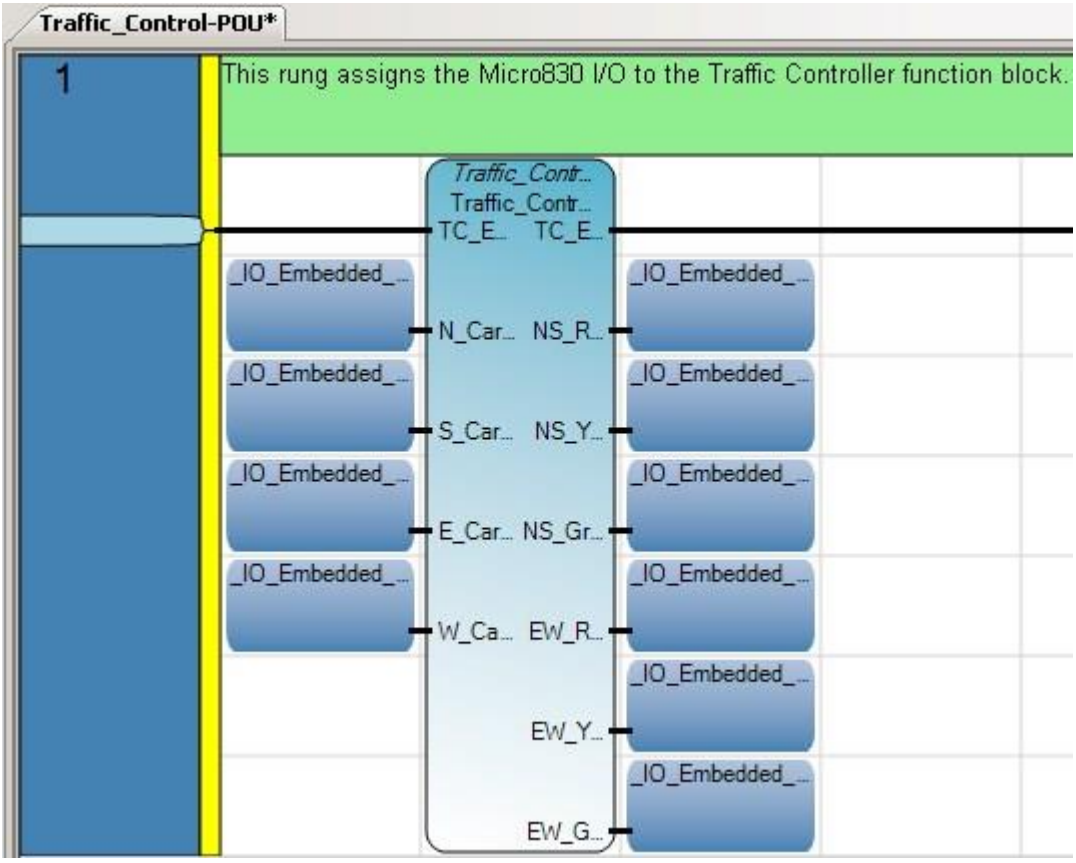
25. Repeat Step 21 for the remaining three input variable blocks, assigning **_IO_EM_DI_1** to **S_Car…**, **_IO_EM_DI _3** to **E_Car…**, and **_IO_EM_DI _4** to **W_Car…** (Note that we skipped using Input 2)



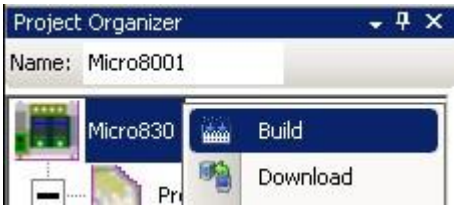
26. The first function block output that connects directly to the ladder rung is the function block enabled bit – it reflects the status of the input enable bit. The remaining six Traffic_Controller function block outputs are “real world” outputs that connect to the red, yellow and green traffic signal lights for each direction. These outputs get mapped to six Boolean output variables local to the function block: **NS_Green_Light, NS_Yellow_Light, NS_Red_Light, EW_Green_Light, EW_Yellow_Light, and EW_Red_Light**. Assign the first six Micro850 digital outputs to the output variable blocks starting With **__IO_EM_DI _0** to **NS_R…** and ending with **_IO_EM_DI _5** to **EW_G…**:

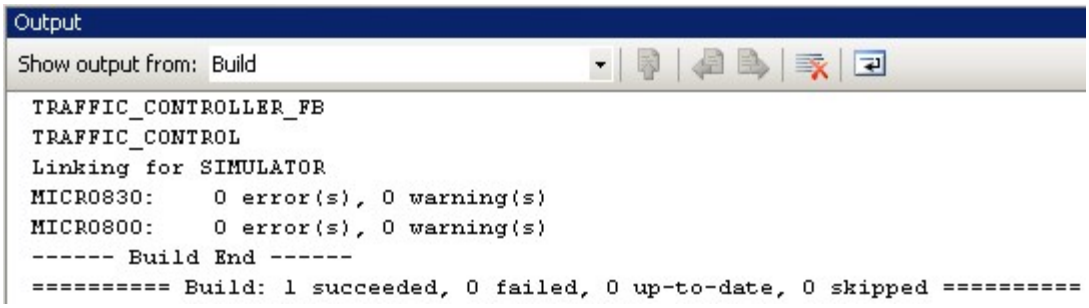


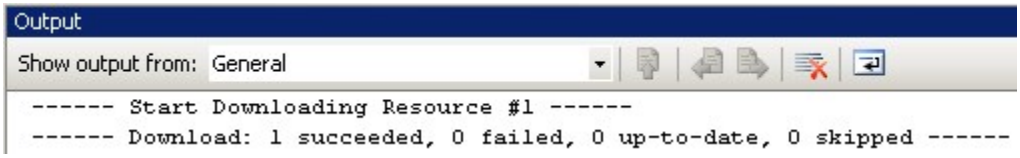
27. Our rung is now complete except for a description of what the rung does. Double click on the green area just above the rung and type in "**This rung assigns the Micro850 I/O to the Traffic Controller function block.**":



28. We are now ready to build and test out our one-rung program. Right click on the Micro830 icon in **Project Organizer** and select **Build**:



29. You should get verification in the **Output** window at the bottom of the screen that the build succeeded (if your build failed, get your lab instructor to review your work thus far):
30. 

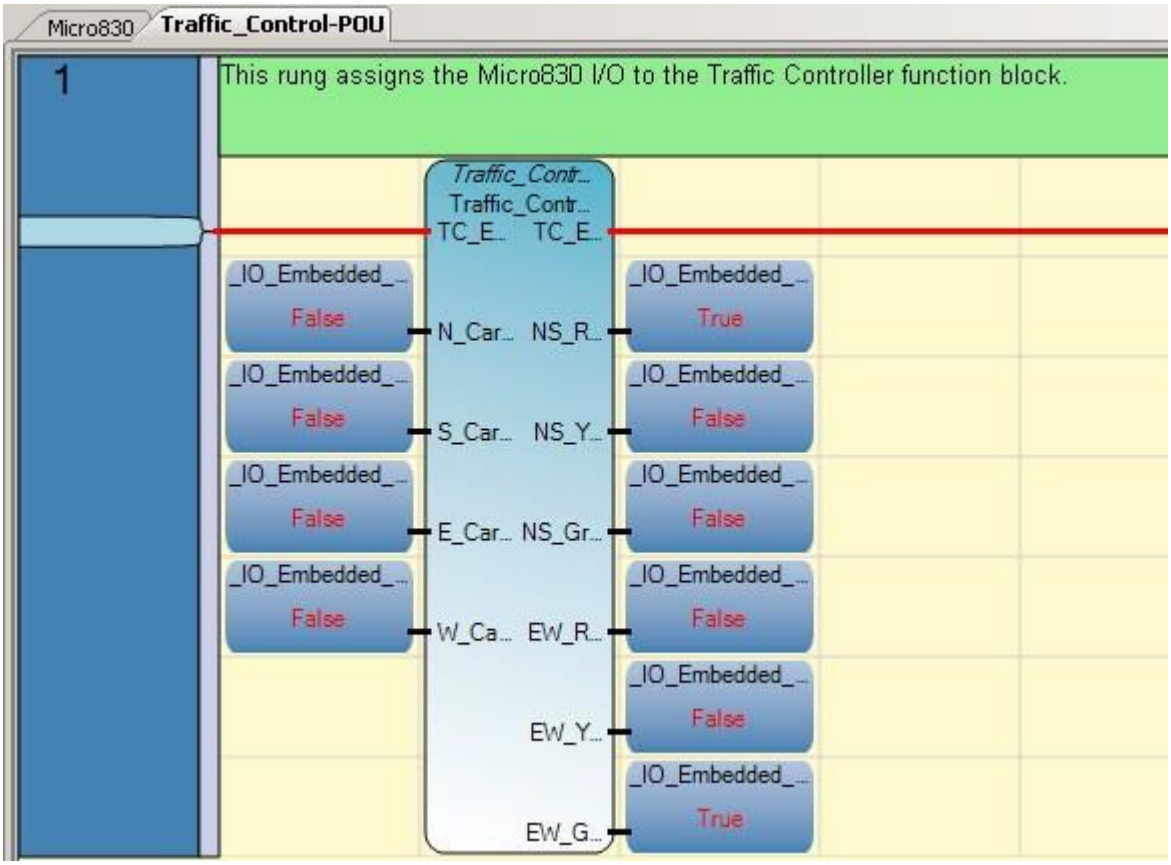
31. Once you have successfully built the project, you are ready to download. Right click again on the Micro830 icon in **Project Organizer** and select **Download**.
32. You should get verification in the **Output** window at the bottom of the screen that the download succeeded (if your download failed, get your lab instructor to review your work thus far):


33. When asked "**Do you want to change to Run mode?**", click **Yes**:


34. Next we will monitor the running program. Click on the **Traffic_Control-POU** tab to view the one-rung ladder program and then click the **Start Debugging** icon 

on the tool bar.

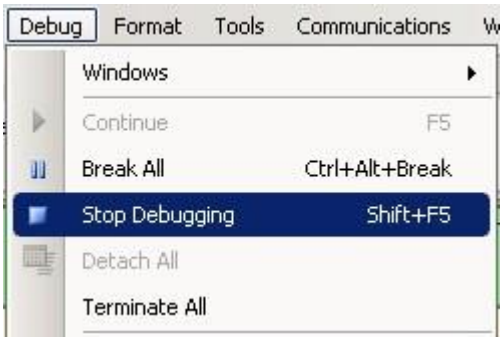
35. In Debugging mode, you are viewing the running program and actual variable values. Your rung should look similar to the following. When the program first starts, the North/South (NS) traffic light is red and the East/West (EW) traffic light is green.:



36. The first thing that you notice is that the rung is red, indicating that the rung is true (power is flowing). The real time state of each of the Boolean inputs (DI0, DI01, DI03, DI04) and outputs (DO0-DO5) is displayed as either True (on) or False (off).

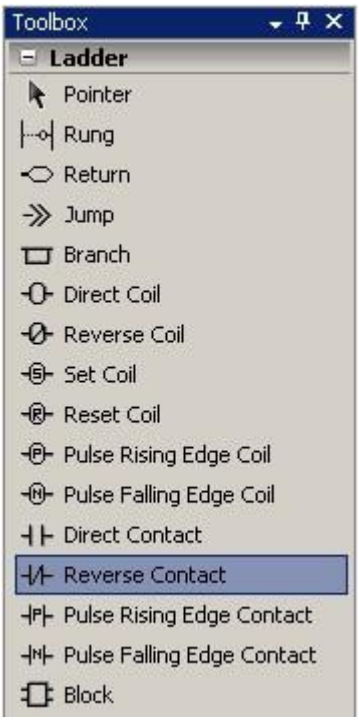
37. If you momentarily push and release either the N_Car_Sensor input button (DI0) or the S_Car_Sensor input button (DI1), you should see the EW traffic light output change from green (DO5 True) to yellow (DO4 True) after five seconds, and two seconds later, from yellow to red (DO3 True), at the same that the NS traffic light changes from red (DO0 True) to green (DO2 True). Similarly, once the EW traffic light is red, if you momentarily push and release either the E_Car_Sensor input button (DI3) or the W_Car_Sensor input button (DI4), you should see the NS traffic light output change from green (DO2 True) to yellow (DO1 True) after five seconds, and two seconds later, from yellow to red (DO0 True), at the same that the EW traffic light changes from red (DO3 True) to green (DO5 True).

38. Select **Debug/Stop Debugging** from the menu bar:

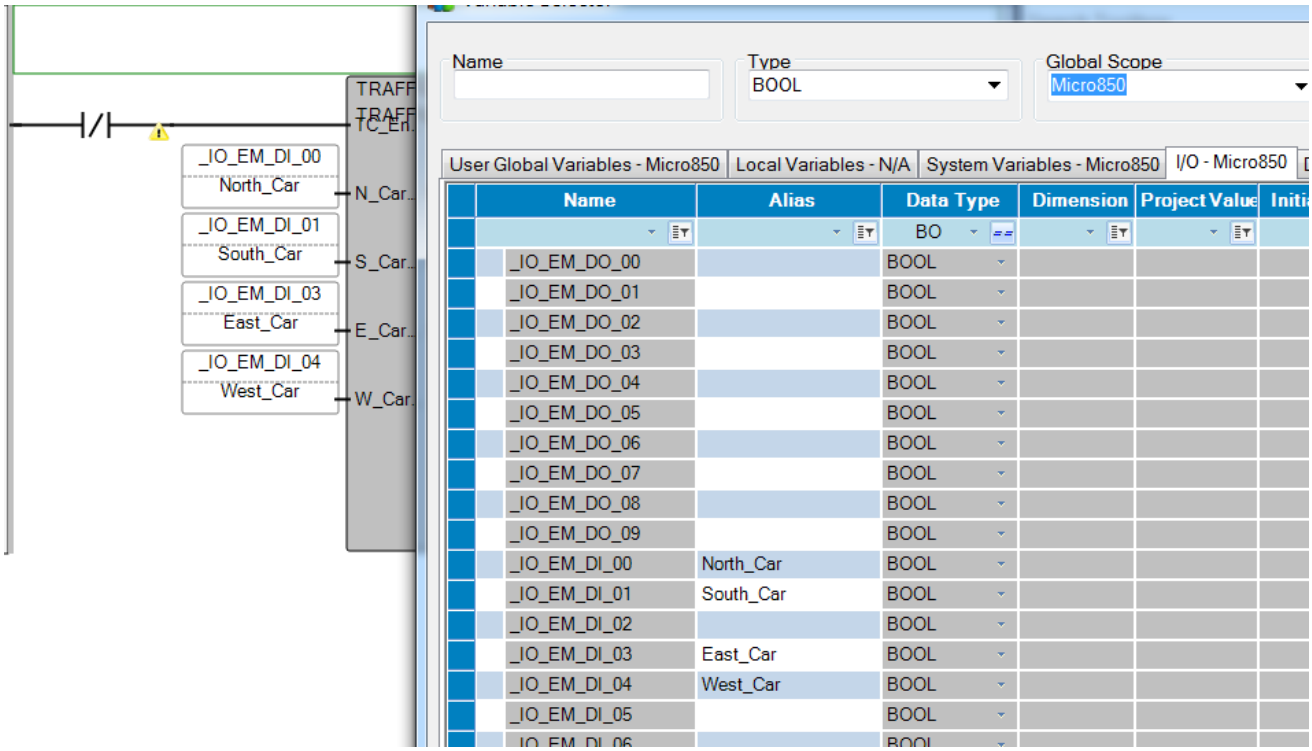


39. Next we are going to modify the ladder to provide a way to flash the lights in the middle of the night. The Micro850 controller has a Real Time Clock plug-in installed in it, we could use the Time of Day instruction to, for instance, flash the lights between 11PM and 5AM. However, to speed things up- let's just simulate a time of day change, so we're just going to use an unused input (DI5) to switch between normal mode and flashing mode.

40. From the **Toolbox** in the lower right-hand screen, click and hold on **Reverse Contact**, drag it onto the rung between the left-hand rail and the Traffic_Controller function block and release:

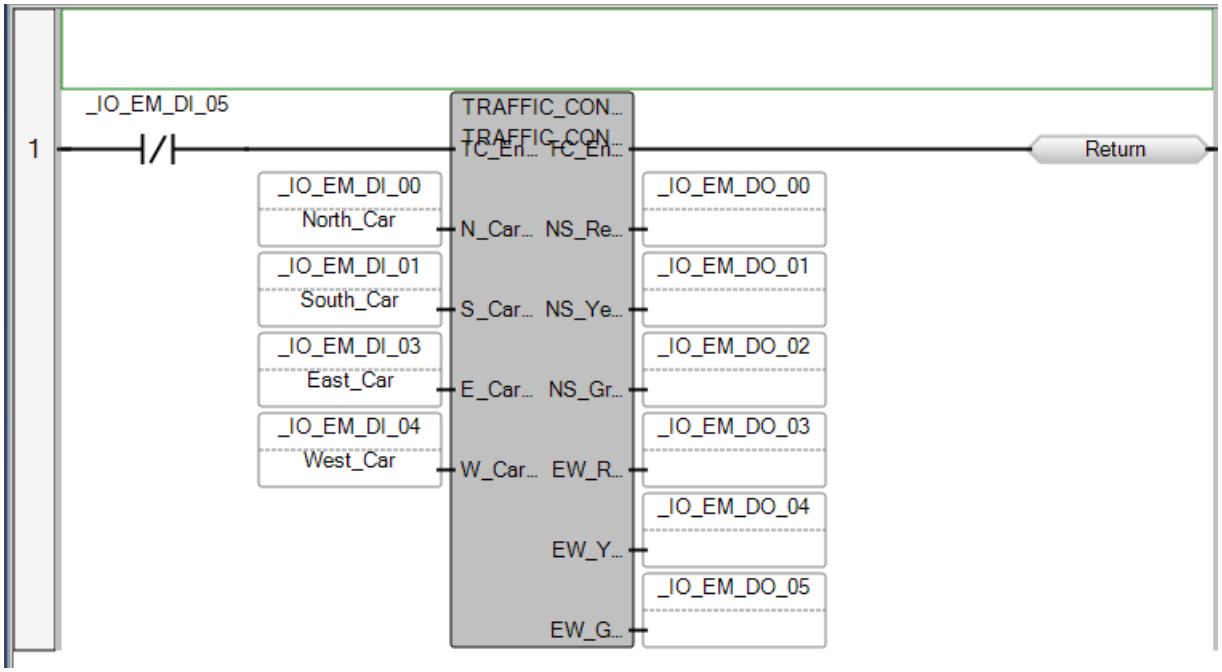


41. When the **Variable Selector** screen appears, select **_IO_EM_DI_5** from the Global Variables and click **OK**:

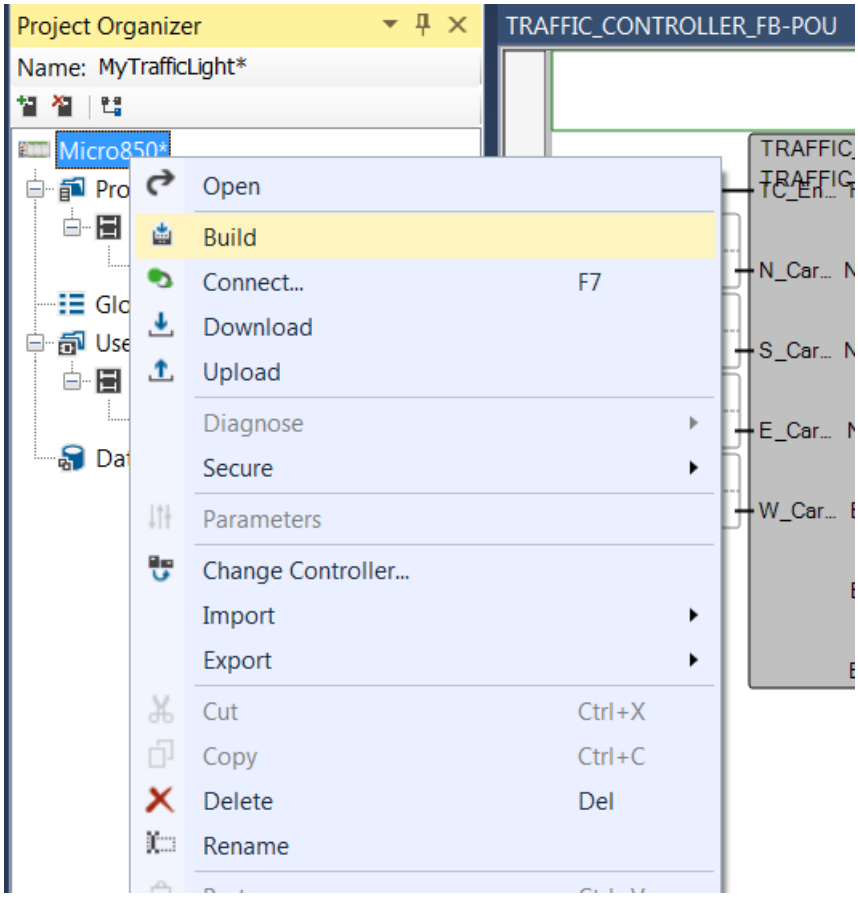


42. Next drag a **Return** from the **Toolbox** onto the rung to the right of the Traffic_Controller function block and release. That way, if the Traffic_Controller function block is enabled, no other rungs of ladder in this file will get executed. If the Traffic_Controller function block is disabled, then the remaining rungs in this ladder program will get executed.

43. Your first rung should now look like this:



Lets go and ahead and rebuild, download and debug the ladder program with this change. Right click on the Micro850 icon in **Project Organizer** and select **Build**:



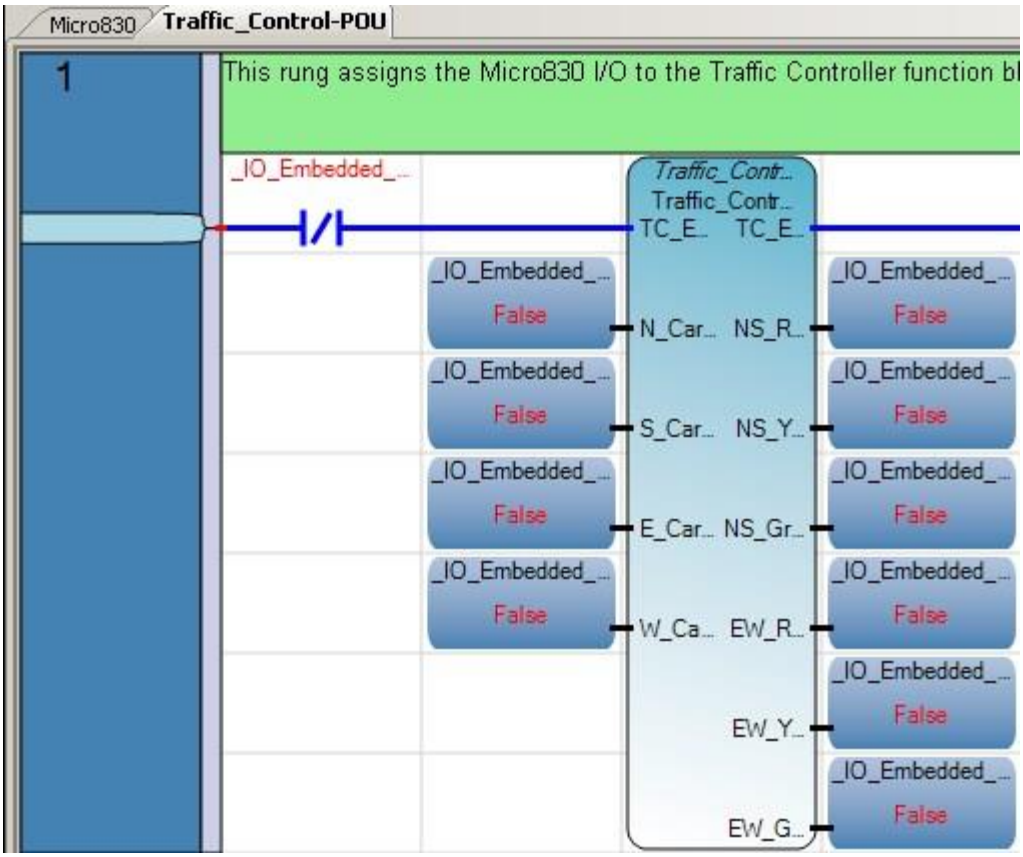
As long as the **Output** window indicates that the build was successful, right click again on the Micro850 icon in **Project Organizer** and select **Download**

Click **Yes** when asked "**Do you want to change to Remote Program mode?**":

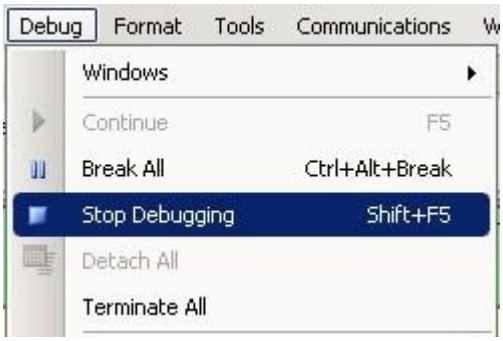
As long as the Output window indicates that the Download was successful, return the controller to Run mode and then click the **Start Debugging** icon 

on the tool bar

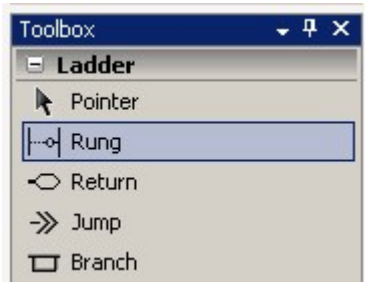
44. Notice that if you push and hold the DI5 input button, the rung color changes from red (power flowing) to blue (power interrupted). When power is interrupted, the function block input enable condition is false, so all of the function block outputs are false as well:

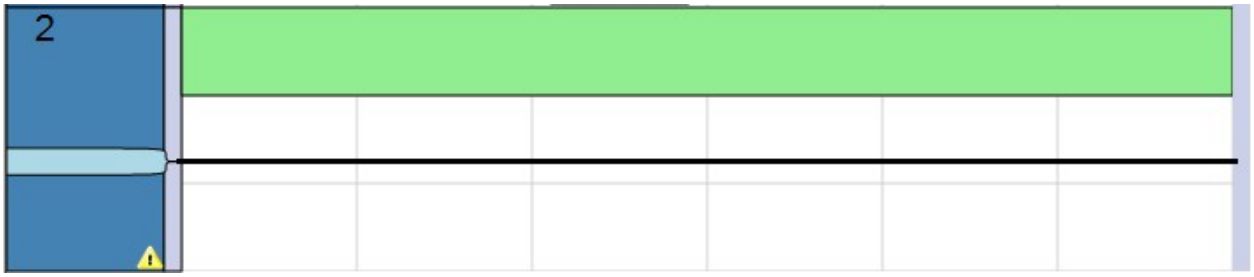


45. Select **Debug/Stop Debugging** from the menu bar:

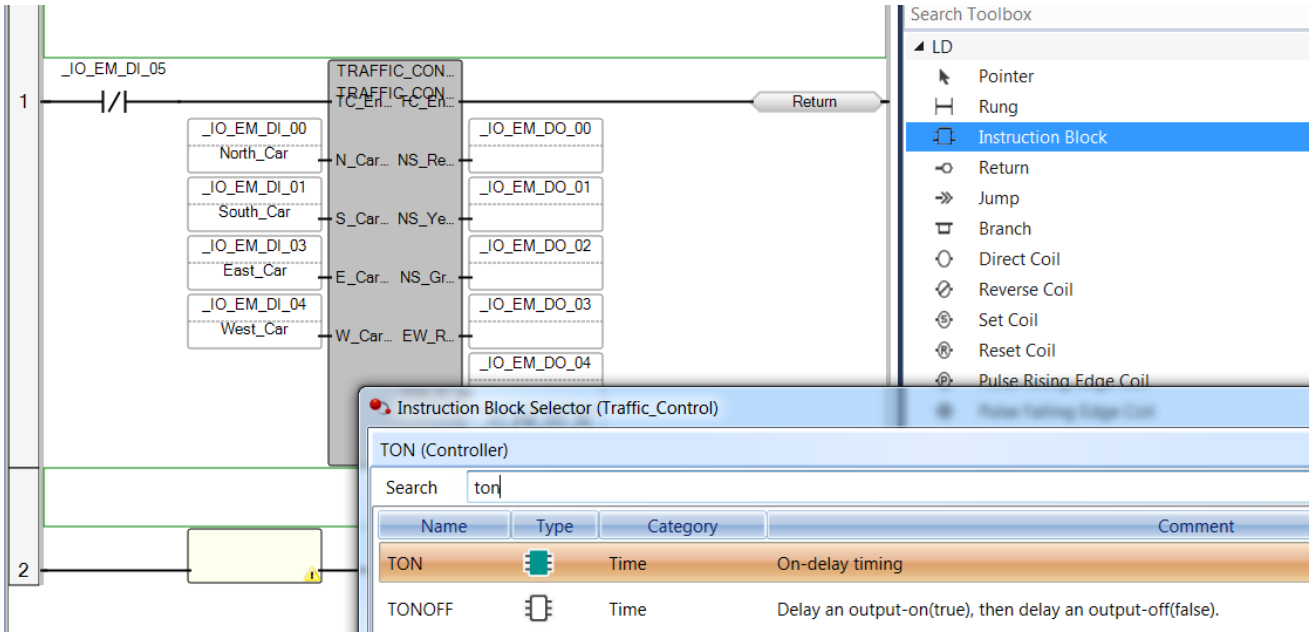


46. Next we are going to add a rung which will flash the traffic lights when the Traffic_Controller function block is disabled (DI5 is on). From the **Toolbox** in the lower right-hand screen, click and hold on **Rung** and drag it beneath the existing rung and release:

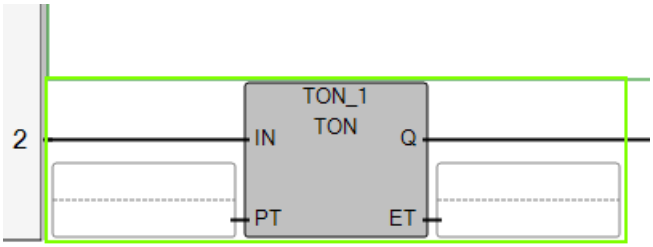




47. We want a 1 second on/1 second off timer to control the flashing of the red lights in the North/South direction, and the yellow lights in the East/West direction. Drag a **Block** from the **Toolbox** and place it on the new rung. When the **Block Selector** screen appears, enter in "to" for the name, select **TONOFF** for Timer On-Off and click **OK**.

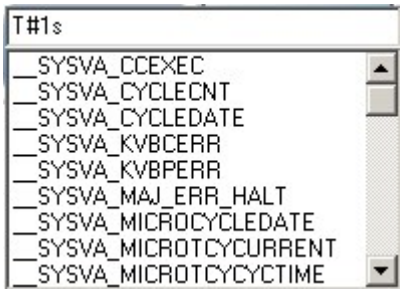


Your rung should look like this so far:



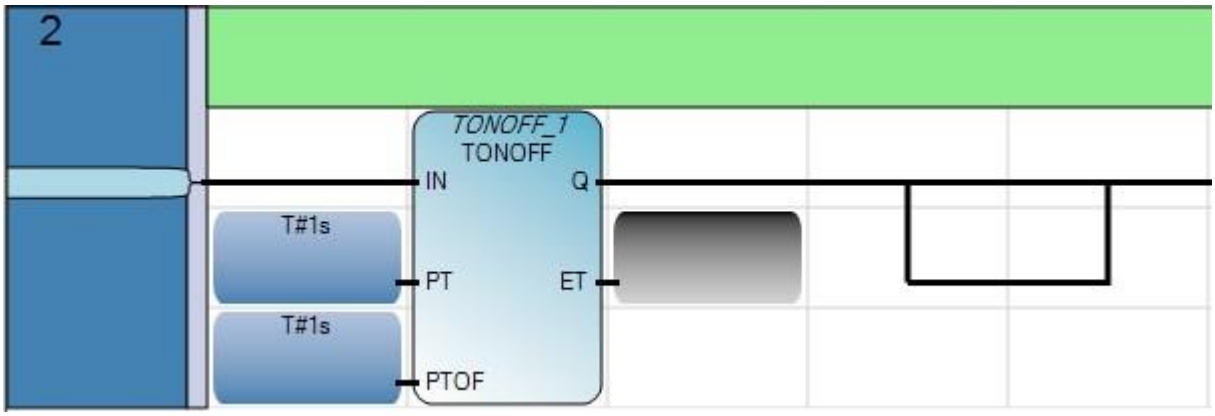
48. As long as the Input (**IN**) to the **TONOFF** is true, then the Elapsed Time (**ET**) begins timing until the Elapsed Time equals the Preset Time (**PT**), at which time the Output (**Q**) becomes true. When the **IN** goes false, the **ET** resets and begins timing until the **ET** equals the Preset Time Off (**PTOF**), at which time the **Q** goes false.

    We want this timer to be a 1second on/1-second off timer, so click in the top of the **PT** input block and enter in "**T#1s**", where **T#** means this is a Time constant and **1s** indicates 1 second:

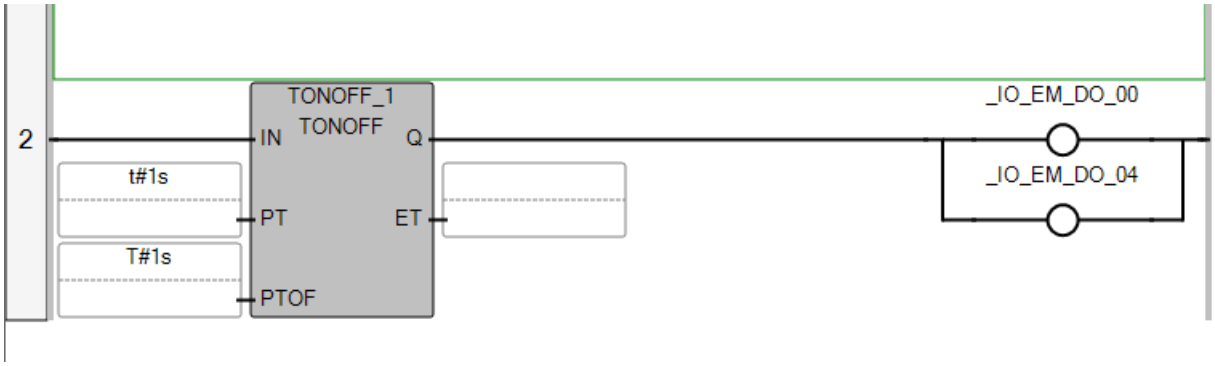


Repeat for the PTOF input block.

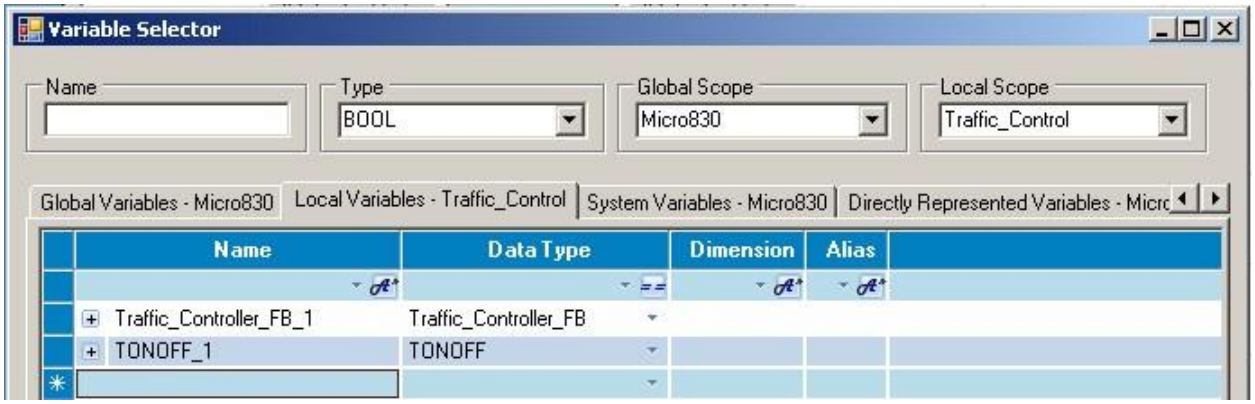
49. When the on-timer has timed out and the output **Q** becomes true, we want the NS_Red_Light (DO0) to turn on, as well as the EW_Yellow_Light (DO4). Since we have two outputs that we want controlled off of **Q**, we need a parallel branch on the rung. Drag a **Branch** from the **Toolbox** onto the rung just right of the **TONOFF**:



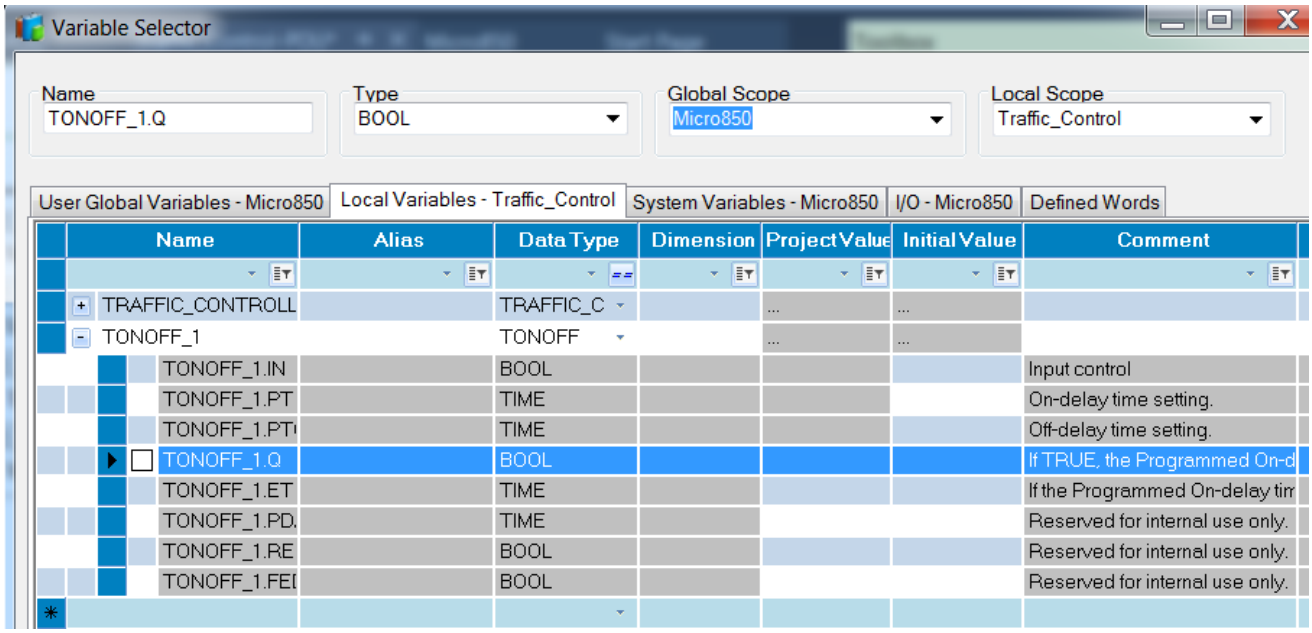
50. We want both DO0 and DO4 to be controlled by **TONOFF** output **Q**, so drag a **Direct Coil** from the **Toolbox** to the upper branch and assign variable **_IO_EM_DO_0** to it. Then Drag a Ranch instruction below that and drag a second **Direct Coil** from the **Toolbox** to the lower branch and assign variable **_IO_EM_DO_4** to it:



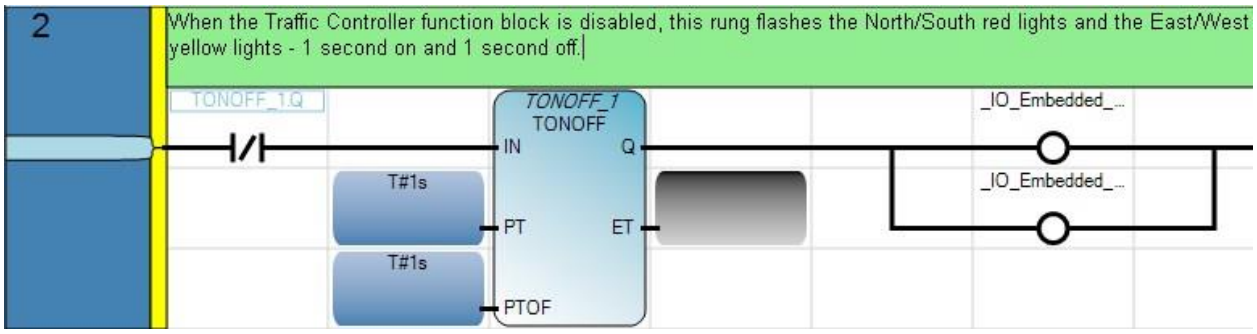
51. Next, we need a way turn the **IN**put off once **Q** turns ON in order to trigger the OFF timer. Therefore, we will add a **Reverse Contact** to the left of the **TONOFF IN** that is controlled by the **TONOFF Q** bit. Drag a **Reverse Contact** to the left of the **TONOFF** block. When the **Variable Selector** screen appears, click on the **Local Variables – Traffic_Control** tab. If no names appear in the **Name** column, click on the cell to the right of the "*****"and you should see the following:



52. Click on the "**+**" to the left of **TONOFF_1** to expand its variables. Select **TONOFF_1.Q** and click **OK**:



53. Document the rung by double clicking in the green space and typing "**When the Traffic Controller function block is disabled, this rung flashes the North/South red lights and the East/West yellow lights – 1 second on and 1 second off.**" Your rung 2 should look like this:



54. **Build** and **Download** the updated program. With the controller back in **Run** mode, test the program out. As long as input button DI5 is not pressed, the program should work as before. If you press and hold input button DI5, the traffic lights should flash on and off (outputs DO0 and DO4) once a second.

When the input of DI5 is in the state of OFF, the Normally Closed contact will run the function block and with the 'RETURN' statement at the end of the rung, no other code will be scanned.

When DI5 is ON, the normally closed state of the contact opens, the function block and return statement stop and the timer circuit runs.

Take a moment to look at the code for the Traffic Light Function block.

User Defined Function Blocks are a powerful way to code an operation that is used many times over, or used many times simultaneously.